simpleSTORM: A self-calibrating single-molecule localization algorithm

Hamprecht Lab, HCI/IWR

Heidelberg, Germany

2013

Contents

1 Introduction

Conventional optical microscopy suffers from Abbe's resolution limit, which states that objects closer than about half the used wavelength can not be resolved. To be precise, the optical resolution limit is defined as the full width at half maximum (FWHM) of the objective's point spread function (PSF), which can be expressed as $\frac{\lambda}{2 \cdot NA}$ within the focus plane, with λ being the wavelength and NA the numerical aperture of the objective used, typically about 1-1.5. Several techniques have been developed to overcome this limitation. The most widely used include STORM[?] (stochastic optical reconstruction microscopy), dSTORM[?] (direct STORM) and PALM[?] (photoactivated localization microscopy). All of these are commonly known as single-molecule localization microscopy and are based on the common principle of taking multiple images per sample, with only a small subset of fluorophores being excited at each frame. The objective's PSF can then be fitted to the individual intensity distributions, yielding a precise molecule localization with a resolution of about 20 nm[?].

Although numerous software applications for single-molecule localization have been published[? ? ?], these usually offer a plethora of configuration possibilites, raising the entry threshold for the localization microscopy novice. simpleSTORM aims to provide good localization results with virtually no user input while still allowing manual fine-tuning of the necessary parameters.

2 Installation

2.1 Prerequisites

The R software package[1] has to be installed and the directory of the R executable included in the PATH environment variable.

[1] `http://www.r-project.org`

2.2 Binary packages

Binary packages for Windows and MacOSX >= 10.7 are provided in form of simple installers.

2.3 Compiling from source

2.3.1 Dependencies

- A recent C++ compiler with C++11 support (Any GCC version >= 4.7 will do the job)
- VIGRA and its dependencies
- libfftw in both single and double precision
- librudeconfig
- R
- libhdf5 (optional)
- Qt 4 >= 4.6
- CMake >= 2.8

2.3.2 Compiling

Compilation follows the standard procedures. After running CMake, specifying the paths to the dependencies manually if required, simpleSTORM can be compiled, e.g. by running `make` on GNU/Linux, MacOSX, or Windows when using MinGW. Installer packages on Windows and MacOSX can be built by issuing the `make package` command. When using MinGW on Windows, be sure to use a threading-capable version[2]. When using Clang on MacOSX, the Clang standard C++ library has to be used for C++11 suppport. This can be accomplished by adding `-stdlib=libc++` to `CMAKE_CXX_FLAGS` in CMake.

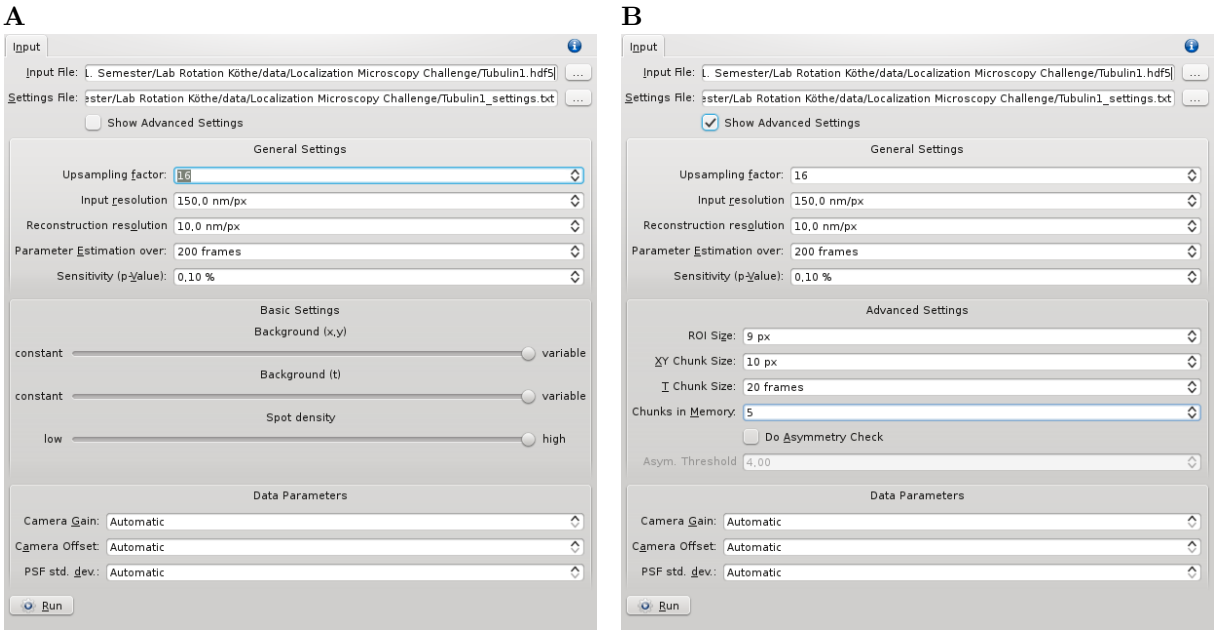

Figure 3.1: The input window of simpleSTORM. **??** The basic view for users not familiar with the algorithm. **??** The advanced view for expert users.

3 GUI manual

3.1 Input

simpleSTORM can process TIFF, SIF and HDF5 files, the path to the input file can be entered manually, selected by clicking on the button next to the "Input File" field or the file can simply be dragged into it. If the file is valid, the rest of the buttons and fields will be enabled. Additionally, a settings file can be specified where the selected and estimated parameters will be saved to. If the specified settings file already exists, its contents will be used to preset the current values.

Additional parameters that can be specified are:

Upsampling Factor Interpolation factor for the last stage of the algorithm. A higher value improves precision, but is computationally more intensive.

Input resolution Resolution of the input images in nm/px. Used to determine the resolution of the reconstructed image.

Reconstruction resolution The desired resolution of the reconstructed image in nm/px. Note that this can not be lower than $\frac{\text{upsampling factor}}{\text{input resolution}}$.

[2] `http://sourceforge.net/projects/mingwbuilds/`

Parameter estimation over Estimation of camera gain and offset as well as the PSF width will be performed over only the first N frames, where N is the value in this input field.

Sensitivity (p-value) The threshold for mask computation. A higher value will result in more detections, but also more false positives (i.e. higher recall with lower precision), while a lower value will increase the precision but decrease recall.

Background (x,y) (simple mode only) Background variability in space. This influences the chunk size in X and Y. A higher setting will result in smaller chunks.

Background (t) (simple mode only) Background variability in time. This influences the chunk size in t. A higher setting will result in smaller chunks

Spot density (simple mode only) Density of fluorescence spots. A higher value will increase the asymmetry threshold, allowing merged spots to be detected, but also potentially increasing the false positive rate, while a lower value will result in a lower asymmetry threshold, removing asymmetrical spots.

ROI Size (advanced mode only) Size of the ROI used for upsampling in pixels. Also influences ROI size for PSF estimation.

XY Chunk Size (advanced mode only) Size of chunks in space in pixels. Lower values are suitable for highly dynamic background.

T Chunk Size (advanced mode only) Size of chunks in time in pixels. Lower values are suitable for highly dynamic background.

Chunks in Memory (advanced mode only) Number of chunks to keep in memory simultaneously. A higher value will improve background interpolation at the cost of increased memory consumption.

Do Asymmetry Check (advanced mode only) Whether to filter false positives by asymmetry.

Asym. Threshold (advanced mode only) Threshold of the Hessian eigenvalues ratio for asymmetry detection. A higher value will allow for asymmetrical spots to be detected.

Camera Gain Camera gain if known. Will be estimated automatically if not supplied.

Camera Offset Camera offset if known. Will be estimated automatically if not supplied.

PSF std. dev. Standard deviation of the point spread function. Will be estimated automatically if not supplied.

3.2 Results

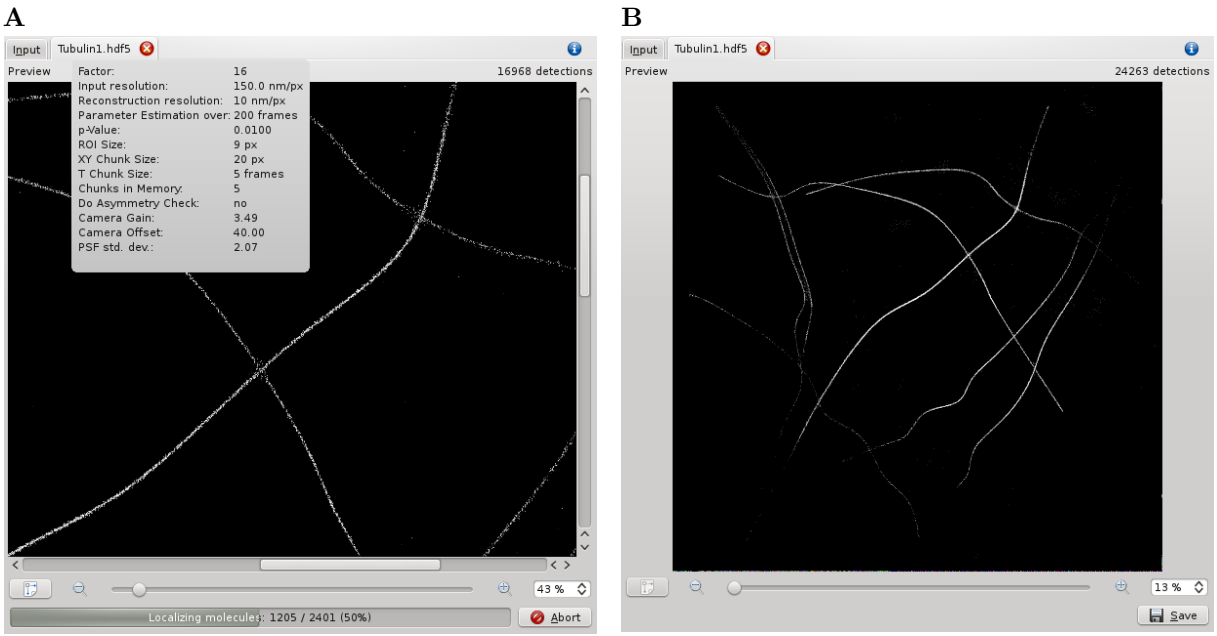

Figure 3.2: After supplying suitable input and hitting the run button, a new tab with the result will be opened. **??** While computation is in progress, a progress bar will be displayed. **??** The result can be saved after the computation is finished.

After hitting the Run button, a new tab showing the progress and a live preview will be opened. Several jobs can run simultaneously, the parameters of a job can be shown by hovering with the mouse over the respective tab (**??**). After the computation is completed (**??**), the results can be saved as a text file with the localizations, a high-resolution image (32-bit TIFF, can be opened with ImageJ), or both.

4 Commandline Interface (CLI) manual

4.1 Synopsis

storm [OPTIONS] input-file

4.2 Description

`-?, --help` Print help message and exit.

`-v, --verbose` Print debug messages during localization.

-V, --version Print version information and exit.

-g factor, --factor=factor The upsampling factor. Default: 8.

-P frames, --cam-param-frames=frames The number of frames to perform parameter estimation over. Default: 200.

-c file, --coordsfile=file The path to the output localizations file. Default: path to input file with .txt extension.

-p size, --pixelsize=size Pixel size in nanometers. Default: 1 (i.e. the localizations will be in pixel coordinates)

-s file, --settings=file Path to settings file. Default: input-file_settings.txt . Parameters used for localization as well as estimated parameters will be stored here and loaded upon re-running simpleSTORM. Parameters without command-line options can be changed here, as well.

-m size, --roi-len=size ROI size in pixels. Default: 9

-a thresh, --asymmetry-threshold=thresh Threshold for asymmetry-based false positive filtering. Default: 4

5 Algorithm

The localization algorithm is subdivided into four stages. In stage 1, the camera gain and offset are being estimated using difference-based noise modeling in the time dimension[?]. After constructing a mean projection over a subset of frames, the algorithm attempts to select pixels over the entire value range, for which the $\mu^{(1)}$ skellam parameter is computed. A linear function is then fitted to the $\mu^{(1)}$/mean value plot, whose slope and root are assumed to be the camera gain and offset, respectively. To account for pixels containing signal in a subset of frames, which generally significantly deviate from the linear relationship between the mean value and $\mu^{(1)}$, fitting is only performed over the most compact subset of data points[?].

A common pre-processing step is applied to stages 2-4, consisting of transformation of the image data to a Poisson distribution using the parameters obtained in stage 1, transformation of the Poisson distributed image into a unit-variance Gaussian using the Anscombe transform[?], and local background subtraction. For the background estimation,

the image stack is subdivided into 3-dimensional chunks, for each chunk the median, which for a Gaussian distribution is identical to the mean, but better suited here due to its resistance to outliers such as pixels with signal, is computed. The low-resolution median image is interpolated to the original resolution using a cubic B-spline and the interpolated image subtracted from the original. To conserve memory, the image stack is processed incrementally, with at most 5 (in the default configuration) t-chunks kept in memory. This strategy results in the preprocessing having to be repeated for each of the three stages where it is applied.

The camera gain estimated in stage 1 is further refined in stage 2 by an iterative algorithm. As an incorrect gain estimate results in a non-Poisson distribution and thus directly affects the variance of the Gaussian distribution after the Anscombe transform, a 1D Gaussian PDF is fitted to the histograms of a subset of frames after pre-processing, the gain adjusted (using the estimate from stage 1 as a starting value) until the mean variance converges to 1. The obtained gain value is used in pre-processing for the subsequent stages.

A good PSF estimate is key to optimal localization results. simpleSTORM assumes the PSF to be a 2D-Gaussian with equal variance in X- and Y-direction. In stage 3 a subset of frames is processed, for each frame several randomly chosen local maxima above a dynamically computed threshold are chosen and the Fourier transform of a region (subsequently referred to as ROI, region of interest) around each maximum is computed. As applying the Fourier transform to a Gaussian signal in the time or spatial domain results in a normally distributed signal in the frequency domain with the variances being inversely proportional to each other (the exact formula is shown in **??**, where N is the ROI width in pixels, σ_s the standard deviation of the Gaussian in spatial domain and σ_f the standard deviation of the Gaussian in frequency domain), a 2D Gaussian PDF is fitted to the mean magnitude image obtained from all frequency spectra and its variance used to compute the PSF variance in the spatial domain.

$$\sigma_s = \frac{N}{2 \cdot \pi \cdot \sigma_f} \tag{5.1}$$

Fluorophore localization happens in stage 4. The pre-processed image is convolved with a Gaussian filter using the variance estimated in stage 3, as this so-called matched

filtering results in an optimal signal-to-noise ratio (SNR). To roughly localize the signal the filtered image is thresholded with a user-supplied p-value under the assumption of a zero-mean, unit variance Gaussian distribution, the threshold being corrected for the filtering. Regions around local maxima with an intensity above the threshold are interpolated using a cubic B-spline and a user-defined interpolation factor, the local maximum of the upsampled ROI is assumed to correspond to the fluorophore coordinate. To reduce false positives a filtering based on spot asymmetry, computed via the eigenvalues ratio of the Hessian matrix at the maximum, can be applied.